NFA Reduction for Regular Expressions Matching Using FPGA

Vlastimil Košař, Martin Žádník, Jan Kořenek
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, Czech Republic
email: {ikosar, izadnik, korenek}@fit.vutbr.cz

***Abstract*—Many algorithms have been proposed to accelerate regular expression matching via mapping of a nondeterministic finite automaton into a circuit implemented in an FPGA. These algorithms exploit unique features of the FPGA to achieve high throughput. On the other hand the FPGA poses a limit on the number of regular expressions by its limited resources. In this paper, we investigate applicability of NFA reduction techniques – a formal aparatus to reduce the number of states and transitions in NFA prior to its mapping into FPGA. The paper presents several NFA reduction techniques, each with a different reduction power and time complexity. The evaluation utilizes regular expressions from Snort and L7 decoder. The best NFA reduction algorithms achieve more than 66% reduction in the number of states for a Snort ftp module. Such a reduction translates directly into 66% LUT-FF pairs saving in the FPGA.**

I. Introduction

The number of rules in Network Intrusion Detection Systems (NIDS)[1] grows each year as grows the number of threats and attacks on the Internet. Many of these rules include Regular Expressions (RE) to match malicious strings in a packet payload. Regular expression matching is also used in other network applications. For example L7 decoder[2] uses RE to identify application protocols or hidden traffic.

Current processors do not provide sufficient processing power to match large sets of regular expressions on multigigabit speeds [3]. Therefore many hardware architectures have been created to accelerate RE matching using FPGA [4], [5], [6]. Nevertheless, these architectures are able to achieve high speed only for small sets of REs due to the limited FPGA resources. The hardware architectures based on Deterministic Finite Automata (DFA) [3], [7], [4] are limited by the size and speed of a memory as the determinization leads to a significant growth of the number of states and transition tables. The architectures based on Nondeterministic Finite Automata (NFA) [8], [5], [9] are limited by the size and capacity of FPGA chips since the transition table is mapped directly into the FPGA logic.

With the growing amount of attacks, worms and viruses, NIDS systems have to match more and more REs. It means that the amount of required FPGA logic increases not only due to the increasing speed of network links but also due to the growth of RE set. It is important to reduce the amount of consumed FPGA logic to support more REs. A significant effort has already been done in this direction. Clark reduced [5] the size of the NFA circuit by a shared character decoder and by a prefix sharing. Lin extended Clark's architecture by infix and suffix sharing [6]. Subsequently, Sourdis improved mapping of NFA to FPGA by better representation of the counting constraint [9] and also improved representation of character decoders. Becchi introduced an optimization of the NFA in [10] which is close to the NFA reduction. The optimization is based on modified subset construction algorithm. The shortcomings of this method is its sensitivity to a placement of epsilon transitions and it does not allow to share suffixes belonging to a single regular expression. Unfortunately no comparison is given with respect to NFA reduction. Moreover, the NFA reduction performs better according to the results in this paper.

Our focus and main contribution is a study of NFA reduction techniques in the field of RE matching in FPGA. In particular,

- we survey several variants of NFA reduction techniques,
- we evaluate NFA reductions on NIDS REs and we provide synthesis results for the FPGA technology,
- we modify the NFA reduction to fit the purpose of RE searching.

The NFA reduction techniques are well known in the research field of formal verification but, to the best of our knowledge, have not been utilized in the field of regular expression matching in the FPGA. The NFA reduction forms a formal aparatus to reduce the number of states and transitions in the NFA. The reduction may easily complement most of the NFA-to-FPGA mapping algorithms and render the ad hoc optimizations unnecessary.

The article is divided into five sections. Brief summary of the NFA reduction algorithms and their utilization for regular expression matching follows the introduction. The NFA reduction algorithms are extended in the third chapter to identify the matched expression. The achieved results are presented in the fourth chapter, followed by the conclusion.

II. NFA Reduction

Regular expression matching on the FPGA may utilize reconfigurability of the FPGA and optimize the hardware architecture to a given set of regular expressions. The circuit is not usually constructed from the regular expressions directly. The expressions are transformed into the NFA and the NFA is mapped into the FPGA. The NFA reduction should be applied in between these two steps to remove a redundancy in the assembled NFA. The redundancy of the NFA automaton is

caused by REs themselves, by the RE to NFA conversion and by the similarity between the RE. Removing the redundancy impacts the size of the circuit implemented in an FPGA directly. The proposed workflow is shown in Fig. 1.

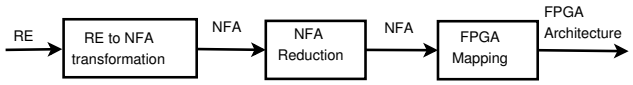

Fig. 1. Workflow with NFA reduction

In case of the DFA the redundancy can be eliminated by minimization algorithm [11].

Since the NFA minimization is a PSPACE-complete problem [12] it is not applicable for large automata which are typically derived from NIDS sets of REs. Therefore many algorithms trade off the time complexity for the degree of the NFA minimization. It is possible to significantly reduce the NFA even in the less than the polynomial complexity. Most of these NFA reduction algorithms have been used to reduce labeled transition systems [13], [14], [15] in formal verification and in a language theory [16], [17], [18], [19]. But we are not aware of any study that applies these algorithms in the field of RE matching engines targeting FPGAs.

The following paragraphs present the reduction algorithms based on an equivalence relation and a preorder relation on a set of NFA states. We pick these two approaches due to their low time complexity but other NFA reduction approaches with higher complexity may be used as well.

The equivalence based reduction algorithm uses left and right equivalence over NFA states. The reduction algorithm merges right (left) equivalent states. The algorithm repeats reduction by left and right equivalence until further state merge is impossible. The time complexity of the algorithm for NFA $M = (Q, \Sigma, \delta, S, F)$ is $O(|\delta|\log|Q|)$ and space complexity is $O(|\delta| + |Q|)$ [18]. For more information see [16], [17], [18], [13].

The preorder based reduction algorithms make use of left and right preorder over NFA states. The reduction algorithms combine left and right preorders in various ways to identify mergeable states. Three different algorithms ware selected. The first algorithm repeats reduction by left and right preorder until further state merge is impossible [20]. The second algorithm uses both left and right preorders simultaneously in each reduction step and repeats those steps until no further reduction is possible [18]. The last one constructs mediated preorder from left and right preorder and the mediated preorder is repeatedly used to determinate mergeable states [15]. Preorder based reduction algorithms are more powerful in terms of NFA reduction then equivalence based [18]. An efficient algorithm is able to compute the preorder relation on NFA in time complexity of $O(|\Sigma||P||Q| + |P||\delta|)$ and space complexity of $O(|\Sigma||P||Q|)$, where P is the size of a final partition [15]. For detailed descriptions see [19], [18], [13], [15].

III. NFA Reduction for Regular Expressions Searching

Many network applications need to know which RE is matched. For example, NIDS rises an alert according to the matched RE, L7 decoder identifies different application protocols by different REs. Information about a matched RE is represented by final states in generated NFA. Unfortunately, the NFA reductions join the final states and the relation between the final states and the REs is lost. The reduced NFA is still able to detect but not to identify which RE is matched.

Therefore we define modified NFA and alter the reduction method to preserve final states together with the relationship to the REs.

We define Multi-language Nondeterministic Finite Automaton (MNFA), which extends the NFA definition by a finite set of labels (corresponding to REs) and mapping of labels to the final states.

Definition 3-1. **Multi-language nondeterministic finite automaton (MNFA)** is seven-tuple $M = (Q, \Sigma, \delta, S, F, L, l)$ where:

- L is a finite set of labels of accepted languages (REs)
- $l \subseteq FxL$ is a function mapping the final states to the labels of accepted languages

The final states are marked by the labels to represent different REs. Those states with the different labels can not be in equivalence relation and can not be joined. This way the algorithm preserves at least one final state for every RE.

We can define an equivalence and preorder relations on MNFA.

Definition 3-2. **Right and left equivalence relation on MNFA** M**.** Let $M = (Q, \Sigma, \delta, S, F, L, l)$ be MNFA. Then $\subseteq_R$ is defined as coarsest equivalence relation on Q satisfying:

$$\forall p, q \in F : p \equiv_R q \Rightarrow l(p) = l(q) \quad (1a)$$

$$\equiv_R \cap (F \times (Q \backslash F)) = \emptyset \quad (1b)$$

$$\forall p, q \in Q, \forall a \in \Sigma,$$
$$p \equiv_R q \Rightarrow [\forall q^{'} \in \delta(q, a), \exists p^{'} \in \delta(p, a), q^{'} \equiv_R p^{'}] \wedge$$
$$[\forall p^{'} \in \delta(p, a), \exists q^{'} \in \delta(q, a), p^{'} \equiv_R q^{'}] \quad (1c)$$

Left equivalence relation $\equiv_L$ is symmetrically defined on reverse MNFA.

Definition 3-3. **Right and left preorder relation on MNFA** M**.** Let $M = (Q, \Sigma, \delta, S, F, L, l)$ be MNFA. Then $\subseteq_R$ is defined as coarsest preorder relation on Q satisfying:

$$\forall p, q \in F : p \subseteq_R q \Rightarrow l(p) = l(q) \quad (2a)$$

$$\subseteq_R \cap (F \times (Q \backslash F)) = \emptyset \quad (2b)$$

$$\forall p, q \in Q, \forall a \in \Sigma,$$
$$(p \subseteq_R q \Rightarrow \forall q^{'} \in \delta(q, a), \exists p^{'} \in \delta(p, a), p^{'} \subseteq_R q^{'}) \quad (2c)$$

Left preorder relation $\subseteq_L$ is symmetrically defined on reverse MNFA.

Despite the altered definition it is still possible to calculate these relations with algorithms mentioned in Section II. The only modification to these algorithms is caused by the first condition in the definition (Equations 1a and 2a) which allows the equivalency of final states belonging only to the same RE. The condition affects the calculation of the initial relation. The

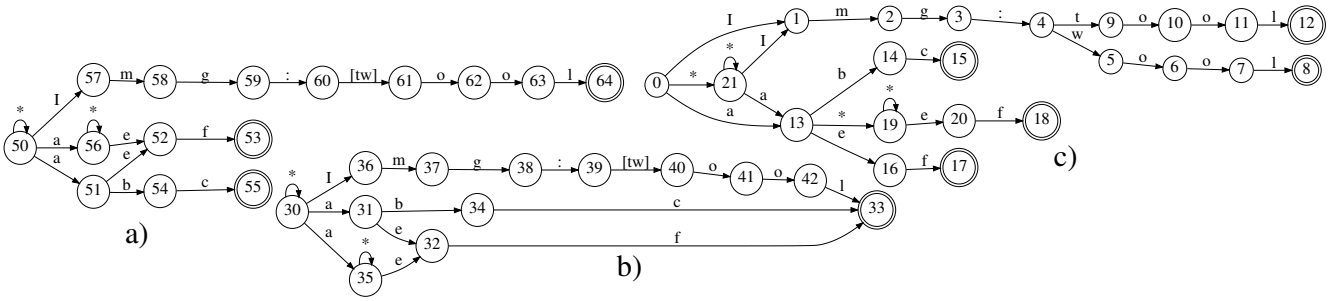

Fig. 2. Reduction of: a) MNFA by equivalence b) NFA by equivalence c) NFA by modified subset construction. The (M)NFA was constructed from REs: `/Img:(tool|wool)/`, `/a.*ef|aef/` and `/abc/`

rest of reduction algorithm remains unaltered. Overall computational complexity of reduction algorithms is not affected by altered definition. The difference between NFA and MNFA reduction is shown in Fig. 2. Both reductions use equivalence relations. It can be seen that the MNFA reduction preserves final states and thus is able to recognize matched RE. Reduced NFA contains one shared final state and can detect only a match. Since more restrictions are introduced for equivalence and preorder relations, the reduction power of the algorithms for MNFA is lower than the reduction power of the algorithms for NFA. This can be seen in Fig. 2, where a reduced NFA consists of 13 states whereas a reduced MNFA consists of 15 states.

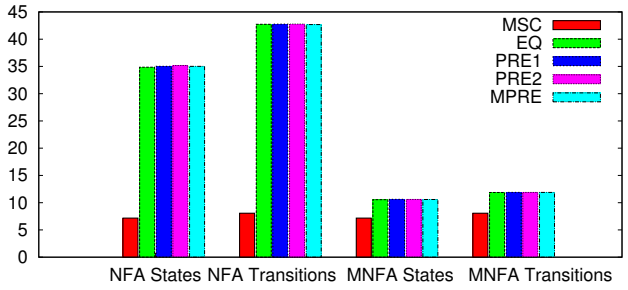

Fig. 3. Reduction of (M)NFA states and transitions for 25 Snort modules, various reduction algorithms.

IV. Evaluation

The algorithms of MNFA and NFA reductions are evaluated on a set of 25 randomly selected modules of the Snort IDS and on the REs available in L7 decoder. Netbench framework [21] is used to evaluate all algorithms. First, the algorithms are evaluated in terms of reduction of NFA states and transitions. Individual REs are converted into small NFAs and subsequently these NFAs are combined into a single NFA. In case of the modified subset construction (MSC) algorithm the NFA construction is performed as described in [3]. In case of other reduction algorithms, all ϵ–transitions are removed after automata union. Subsequently each reduction algorithm is performed and the results are compared. The same procedure is also used to evaluate the MNFA reduction algorithms.

Fig. 3 shows the results of average NFA and MNFA state reduction for all 25 Snort sets. It can be seen that the reduction algorithm based on the MSC is among the least efficient algorithms. We can also see that the NFA reduction algorithms based on the equivalence (EQ), repeated use of preorders (PRE1), simultaneous use of preorders (PRE2) and mediated preorders (MPRE) offer almost the same reduction power despite the different time complexity and the fact that preorder based algorithms have in theory better reduction power than the equivalence ones. As can be seen in Fig. 3, the algorithms exhibit similar results in terms of transitions as well.

We also estimate the amount of utilized FPGA resources which is needed to map the reduced NFA and MNFA into the FPGA. We use one of the basic NFA mapping methods and two of the advanced methods. The basic one is Clark's [22] method with shared decoder of characters and character classes, the first advanced one is Sourdis' [9] with shared character classes, blocks of constrained repetitions and static subpatterns and the second advanced one is at-most two-hot encoding (AMTH) [23]. The amount of lookup tables (LUT), flip-flops (FF) and LUT-FF pairs after Place&Route (PAR) for all three methods is given in Table I. We used Xilinx Virtex-5 155 LXT FPGA for the evaluation. The NFA and MNFA reduction is carried out by algorithm which achieves the best results in previous experiments. Columns NFA and MNFA indicate achievable reduction of LUTs, FFs and LUT-FF pairs using the NFA and MNFA reduction algorithms. It can be seen that significant reduction of utilized FPGA resources is achieved for the basic Clark's method of mapping. For more sophisticated Sourdis' method the reduction is lower but still quite high in some cases. The results demonstrate how the NFA reduction complements the optimization of the mapping algorithms. The impact of NFA and MNFA reductions on maximal achievable frequency after PAR is depicted in Table II.

V. Conclusions

The paper described several NFA reduction algorithms with polynomial time complexity and provided comparison of these algorithms on the NFAs. According to the results the algorithms provided a significant reduction for particular sets of REs. For example, NFA states were reduced by more than 35% in the NFA generated from 25 Snort modules. The results also showed that the NFA reduction helps to save FPGA resources and is complementary to the mapping optimizations. The NFA reduction joins all final states to a single state. As a result the matched RE can not be identified. Therefore we have defined MNFA automaton and defined MNFA reduction which is able to preserve final states. The MNFA reduction provides worse results than NFA reduction which is in line with the expectation. Nevertheless, NFA states are still reduced

TABLE I. REDUCTION OF FPGA LOGIC UTILIZATION AFTER PAR FOR XILINX VIRTEX-5 155 LXT FPGA, VARIOUS NFA AND MNFA REDUCTIONS

	Clark [22]						Sourdis [9]						AMTH [23]					
	NFA			MNFA			NFA			MNFA			NFA			MNFA		
Rule set	LUT [%]	FF [%]	LUT-FF [%]	LUT [%]	FF [%]	LUT-FF [%]	LUT [%]	FF [%]	LUT-FF [%]	LUT [%]	FF [%]	LUT-FF [%]	LUT [%]	FF [%]	LUT-FF [%]	LUT [%]	FF [%]	LUT-FF [%]
L7	15.53	20.36	17.38	8.36	9.86	8.00	13.93	18.54	14.45	8.20	9.72	6.52	21.15	21.11	20.86	9.43	9.55	8.77
backdoor	28.29	29.94	30.02	15.90	15.96	15.56	15.85	18.38	19.26	2.26	2.85	3.42	30.53	31.92	32.09	17.29	18.37	16.68
web-php	40.16	47.66	40.79	20.49	22.81	19.21	39.27	45.89	40.16	20.54	22.47	28.14	33.75	46.99	35.42	16.88	21.05	17.56
ftp	71.00	65.77	65.55	9.78	8.70	8.72	37.18	32.32	34.39	10.29	10.63	8.14	57.25	65.60	57.75	10.45	10.92	10.35
nntp	57.67	58.10	57.76	1.37	1.26	1.33	33.38	25.60	32.28	6.85	4.78	7.31	57.00	57.85	57.20	2.12	1.81	2.05
voip	39.11	47.48	39.73	14.67	14.61	14.17	21.01	39.33	23.80	8.89	11.52	9.57	48.75	38.68	38.68	15.81	17.45	15.35
Snort 25	35.36	37.52	36.60	10.31	10.62	11.32	23.45	26.13	25.67	10.22	11.52	10.69	33.06	37.20	33.77	10.83	11.75	10.60

TABLE II. IMPACT OF (M)NFA REDUCTIONS ON MAXIMAL ACHIEVABLE FREQUENCY AFTER PAR FOR XILINX VIRTEX-5 155 LXT

	Clark [22]		Sourdis [9]		AMTH [23]	
Rule set	MNFA [%]	NFA [%]	MNFA [%]	NFA [%]	MNFA [%]	NFA [%]
L7	7.31	-13.65	7.29	-13.82	0.37	-33.52
backdoor	0.34	3.36	0.00	2.68	-0.27	-0.82
web-php	-10.44	-17.70	-10.44	-17.52	14.41	-10.65
ftp	0.00	-29.17	0.00	-29.17	0.00	-19.19
nntp	0.00	0.00	0.00	0.00	0.00	0.96
voip	0.00	26.37	0.00	26.37	9.21	8.94
Average (Snort 25)	-0.69	-5.14	-0.72	-5.00	0.23	-4.46

by more than 10% for 25 selected Snort modules. We have also evaluated the impact of NFA and MNFA reduction on different mappings algorithms. The result showed that the NFA reduction has a direct impact on the FPGA logic utilization even for highly optimized mapping of NFA to FPGA. The NFA reduction was able to decrease the amount of LUT-FF pairs by 65.55% for Snort ftp module and MNFA reduction was able to decrease the amount of LUT-FF pairs by 28.14% for Snort web-php module.

ACKNOWLEDGEMENTS

This work has been supported by the research programme MSM 0021630528, the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070 and the grant BUT FIT-S-11-1.

REFERENCES

[1] Snort, http://www.snort.org/, 2013.

[2] L7 Filter, http://l7-filter.sourceforge.net/, 2013.

[3] M. Becchi and P. Crowley, "Efficient regular expression evaluation: theory to practice," in *ANCS '08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2008, pp. 50–59.

[4] S. Kumar, J. Turner, and J. Williams, "Advanced algorithms for fast and scalable deep packet inspection," in *ANCS '06: Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*. ACM, 2006, pp. 81–92.

[5] C. R. Clark and D. E. Schimmel, "Scalable pattern matching for high speed networks," in *FCCM '04: Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE Computer Society, 2004, pp. 249–257.

[6] C.-H. Lin, C.-T. Huang, C.-P. Jiang, and S.-C. Chang, "Optimization of pattern matching circuits for regular expression on fpga," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 15, no. 12, pp. 1303–1310, 2007.

[7] Y.-H. E. Yang, W. Jiang, and V. K. Prasanna, "Compact architecture for high-throughput regular expression matching on fpga," in *ANCS '08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2008, pp. 30–39.

[8] R. Sidhu and V. K. Prasanna, "Fast regular expression matching using fpgas," in *FCCM '01: Proceedings of the the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE Computer Society, 2001, pp. 227–238.

[9] I. Sourdis, J. Bispo, J. M. P. Cardoso, and S. Vassiliadis, "Regular expression matching in reconfigurable hardware," *Journal of Signal Processing Systems*, vol. 51, no. 1, pp. 99–121, 2008.

[10] M. Becchi and P. Crowley, "Efficient regular expression evaluation: theory to practice," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '08. New York, NY, USA: ACM, 2008, pp. 50–59.

[11] J. E. Hopcroft, "An n log n algorithm for minimizing the states in a finite automaton," in *The Theory of Machines and Computations*, Z. Kohavi, Ed. Academic Press, 1971, pp. 189–196.

[12] T. Jiang and B. Ravikumar, "Minimal nfa problems are hard," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1991, vol. 510, pp. 629–640.

[13] R. Gentilini, C. Piazza, and A. Policriti, "From bisimulation to simulation: Coarsest partition problems," *Journal of Automated Reasoning*, vol. 31, pp. 73–103, 2003.

[14] L. Tan and R. Cleaveland, "Simulation revisited," in *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS 2001. London, UK: Springer-Verlag, 2001, pp. 480–495.

[15] L. Holík, "Simulations and antichains for efficient handling of finite automata," Ph.D. dissertation, 2011. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=9517

[16] L. Ilie and S. Yu, "Algorithms for computing small nfas," in *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*, ser. MFCS '02. London, UK, UK: Springer-Verlag, 2002, pp. 328–340.

[17] ——, "Reducing nfas by invariant equivalences," *Theor. Comput. Sci.*, vol. 306, pp. 373–390, September 2003.

[18] L. Ilie, R. Solis-oba, and S. Yu, "Reducing the size of nfas by using equivalences and preorders," in *In Combinatorial Pattern Matching, Jeju Island, South Korea*, 2002.

[19] J.-M. Champarnaud and F. Coulon, "Nfa reduction algorithms by means of regular inequalities," in *Proceedings of the 7th international conference on Developments in language theory*, ser. DLT'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 194–205.

[20] J. Högberg, A. Maletti, and J. May, "Backward and forward bisimulation minimisation of tree automata," in *Proceedings of the 12th international conference on Implementation and application of automata*, ser. CIAA'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 109–121.

[21] V. Pus, J. Tobola, V. Kosar, J. Kastil, and J. Korenek, "Netbench: Framework for evaluation of packet processing algorithms," *Symposium On Architecture For Networking And Communications Systems*, pp. 95–96, 2011.

[22] C. Clark and D. Schimmel, "Efficient Reconfigurable Logic Circuits for Matching Complex Network Intrusion Detection Patterns," in *Field Programmable Logic and Application, 13th International Conference*, Lisbon, Portugal, 2003, pp. 956–959.

[23] S. Yun and K. Lee, "Optimization of regular expression pattern matching circuit using at-most two-hot encoding on fpga," *International Conference on Field Programmable Logic and Applications*, vol. 0, pp. 40–43, 2010.